1-1-2011

Secure message routing in structured peer-to-peer networks

Abdolkarim Hajfarajollah Dabbagh
Ryerson University

Follow this and additional works at: <http://digitalcommons.ryerson.ca/dissertations>

Part of the [Computer Sciences Commons](#)

Recommended Citation

Dabbagh, Abdolkarim Hajfarajollah, "Secure message routing in structured peer-to-peer networks" (2011). *Theses and dissertations*. Paper 691.

This Thesis is brought to you for free and open access by Digital Commons @ Ryerson. It has been accepted for inclusion in Theses and dissertations by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact bcameron@ryerson.ca.

SECURE MESSAGE ROUTING IN STRUCTURED PEER-TO-PEER NETWORKS

by

Abdolkarim Hajfarajollah Dabbagh

Bachelor of Science, University of Tehran, Iran, 2005

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2011

©Abdolkarim Hajfarajollah Dabbagh 2011

Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Secure Message Routing in Structured Peer-to-Peer Networks

Master of Science 2011

Abdolkarim Hajfarajollah Dabbagh

Computer Science

Ryerson University

Due to the lack of a centralized server in “Peer-to-Peer” (P2P) networks, users are responsible for the security of these networks. One of the security issues in P2P networks is the security of the message routing. Messages could be altered or modified by attackers while being routed. The conventional security method to avoid this has been “Public Key Cryptography” (PKC). To avoid the certificate management issue in PKC, “Identity-based Encryption” (IBE) has been suggested in which any arbitrary string could be used as a public key. Since IBE is a computationally expensive method, current proposed IBE-based methods are not effective in the message routing phase in P2P networks and highly affect the performance of message delivery time in these networks. This thesis proposes two IBE-based protocols that can be applied effectively to the message routing phase of structured P2P networks, yet provide a satisfactory message delivery time performance. Both protocols benefit from Identity-based key exchange scheme and, therefore, none of them impose any extra communication on the network to secure message routing. Protocol 1 significantly improves the performance of message delivery time compared to the current IBE-based proposed methods. Protocol 2, which requires nodes to store data, has a performance similar to the situations in which no security measures are applied for message routing.

Acknowledgements

I wish to express my gratitude to all those who gave me the possibility to complete this thesis. I am deeply indebted to my supervisor Prof. Dr. Ali Miri whose help, stimulating suggestions and encouragement helped me during this process. I also wish to especially thank my supervisor Prof. Dr. Abdolreza Abhari for his continued guidance and stimulating support.

My colleagues from the Department of Computer Science at Ryerson University supported me in my research work and I thank them for all their assistance and interest.

I would like to extend my appreciation to Rambod Rasti who introduced me to the field of cryptography.

I would also like to take this opportunity to thank my beloved family for all their help during my study.

Last but not least, I would like to thank Mateen, my dearest wife, for all her encouragement and support.

To my son, Iman.

Contents

1	Introduction	1
1.1	Overview	1
1.2	Scope and Goal	5
1.3	Contribution	7
1.4	Thesis Outline	8
2	Related Works	9
2.1	Cryptography	9
2.2	Identity-based Encryption	12
2.2.1	Applications for IBE	13
2.2.2	Drawback	15
2.2.3	Preliminaries	15
2.2.4	IBE Algorithms	17
2.2.5	Identity-Based Non-Interactive Key Exchange	20
2.3	Peer-to-Peer Networks	21
2.3.1	Chord	25
2.3.2	Message routing in structured P2P networks	30
2.4	Identity-based Encryption in P2P Networks	32
2.5	Summary	38

3	Methodology	39
3.1	Introduction	39
3.2	Proposed Methodology	40
3.2.1	Protocol 1	43
3.2.2	Protocol 2	46
3.2.3	Memory-Speed Trade-Off	48
3.3	Analysis	50
3.4	Summary	54
4	Evaluation and Results	55
4.1	Implementation	55
4.2	Environment	56
4.3	Experiments	57
4.3.1	Experiment 1: No Security	57
4.3.2	Experiment 2: IBE	58
4.3.3	Experiment 3: Protocol 1	59
4.3.4	Experiment 4: Protocol 2	60
4.4	Discussion	61
4.5	Summary	71
5	Conclusions and Future Work	73
5.1	Conclusions	73
5.2	Future Work	75
	References	83
	Glossary	85

List of Tables

3.1	Notations used in the methodology.	42
4.1	Settings used in this work's experiments.	56
4.2	Comparison of message delivery time performance of all methods against no security method.	71

List of Figures

2.1	General diagram of IBE	13
2.2	P2P architecture against client/server architecture	22
2.3	General ring topology in Chord [1]	27
2.4	Network topology after insertion of a node	28
2.5	Chord: Search process for key 5 from node 15	30
3.1	Simple search procedure in Chord [2]	41
3.2	Protocol 1	45
3.3	Key-Table structure in protocol 2	47
3.4	Key-Table structure in memory-speed trade off method for protocol 2 . .	50
4.1	Message delivery time with no security methods.	58
4.2	Message delivery time using IBE-based algorithms.	59
4.3	Message delivery time using protocol 1.	60
4.4	Message delivery time using protocol 2.	61
4.5	Comparison of average message delivery time between IBE-only and no security methods.	62
4.6	Comparison of average message delivery time between IBE-only method and no security method for each hop.	62

4.7	Comparison of average message delivery time between IBE-only method and protocol 1.	63
4.8	Comparison of message delivery time between IBE-only method and protocol 1 for each hop.	64
4.9	Comparison of message delivery time between protocol 1 and protocol 2.	64
4.10	Comparison of message delivery time between protocol 1 and protocol 2 for each hop.	65
4.11	Effect of Key-Table size on average message delivery time in protocol 2.	66
4.12	Average number of keys stored in Memory-Speed method of protocol 2.	66
4.13	Comparison of average message delivery time when increasing the number of messages in protocol 2.	68
4.14	Comparison of message delivery time in all methods.	69
4.15	Comparison of message delivery time in all methods for each hop.	70

List of Algorithms

3.1	Protocol 1 - Encrypt	44
3.2	Protocol 1 - Decrypt	44
3.3	Protocol 2 - Encrypt	48
3.4	Protocol 2 - Decrypt	49
3.5	Memory-speed trade off method for protocol 2 - Encrypt	51
3.6	Memory-speed trade off method for protocol 2 - Decrypt	52

Chapter 1

Introduction

In this chapter, the scope of this work will be presented. The goal of this work and its underlying motivation will also be discussed, followed by the outline of the thesis.

1.1 Overview

“Peer-to-Peer” (P2P) networks have become very popular especially with the rapid growth of the Internet. The conventional network architectures are based on the traditional client-server (centralized) model. Despite the relative ease of its implementation, client-server architecture has various disadvantages, such as the server load and bandwidth bottleneck. By having a large number of users, these critical problems become even more apparent. Therefore, modern approaches for network architecture, such as using P2P, or decentralized networks, have been proposed. In P2P architecture, the network functionality is pushed to the users, not the server. As a result, by having a larger number of users in a network, the network will perform more efficiently. P2P networks could be categorized into two main types: “Structured” and “Unstructured”. In the case

of the former, because of the existence of a server-like node (usually at joining level), information about the nodes and the content of the network is known to the users. This is in contrast to the latter, in which nodes do not have such information and therefore, algorithms used for finding resources and routing messages are usually different in these networks. In centralized networks, if the server goes down the whole network will go down. In P2P systems, however, this is not the case. In other words, if a single node goes down the network will still be available for other peers to utilize. This supports the argument that because P2P networks are resilient to node failure, the P2P networks may offer a more robust and reliable alternative.

As previously mentioned, the functionality in P2P networks is pushed to the users. In other words, the users will become responsible for the content (resources being exchanged among users) of the network. No entity is monitoring nodes' behavior in the network and anyone can easily join and leave the network as desired. Accordingly, it is the responsibility of users to spot any security threats. Performing these tasks without having a centralized server seems a much harder task to tackle for this network architecture. The reason for this is that the centralized servers use dedicated tools and technologies in order to protect the system. Therefore, security considerations play a very crucial role in designing P2P networks.

There are many security considerations in P2P networks because they are prone to different attacks such as "Sybil attack" [3] (will be discussed in more details in chapter 2). One of the most challenging security issues in P2P networks is "Message Routing" which is because the users are responsible for delivering the messages themselves. Since users are able to join these networks from any location, the security of the environment - or lack thereof - from which users join the network may be a risk for the network. That is, even if users do not try to attack the network, there might be others who could

overhear the network communication (and be able to alter/modify the messages) but are not members of the network (peers could be located all around the world in P2P networks and they could be targeted by attackers). Therefore, message routing which is solely done by peers in the network could be at risk: when peers search for an item in the network, their queries could be altered by attackers before proceeding to the destination. In this case, the recipient will not realize that the query has been altered. This could result in situations in which an irrelevant result (or no result at all) is provided for a query, which clearly makes the network non-functional.

There are many techniques available to protect messages in P2P networks. “Public Key Cryptography” (PKC) is one of these techniques in which users have their own private keys as well as public keys. If an individual desires to send a message to another entity in the network, he/she can use the recipient public key to encrypt the message. On the other side, the recipient should use his/her private key (no one else should have access to it) in order to decrypt the message. The advantage of this method is that there is no key exchange necessary to send/receive messages. However, this method has its own disadvantages. For example, how to verify a person’s public key? This problem begs the following questions: How can users authenticate other parties’ public keys? What if somebody is pretending to be someone else?

There are some solutions for the above-mentioned problem such as having a “Certificate Authority ” (CA). That is, there should be an authority who issues certificates (containing public key information); an authority who is responsible for authenticating users. This solution has some disadvantages as well, such as the fact that each user may need to store many certificates on his station and this may require a large space for storage as well as a challenging task for managing all the certificates. In addition, if users would like to verify whether a certificate is still valid, they have to contact the

“Revocation Authority” (RA) to check whether the certificate is revoked or not. This often causes a communication overhead for the network. It is worth mentioning that in this scheme, users could join different networks (with different CAs). Therefore, users could be assigned different public keys for each of these networks they join. As a result, other users might have to store multiple public keys for a single user.

Other than the above disadvantages, it should be added that since a search query might be passed to tens of users until it reaches its destination, applying PKC in every step of the search before it reaches its destination could potentially make the users in the network face a huge delay in their search queries (considering the slowness of the PKC and certificate issue as well).

Therefore, there is a significant need to design a robust strategy for search processes (message routing) in P2P networks that is both practical and secure against outside attackers. The nature of P2P networks (large number of users who repeatedly leave and join the network) must be also considered.

In order to tackle the above-mentioned certificate issue in PKC, Shamir [4] introduced the “Identity-Based Encryption” (IBE) scheme. In IBE, users have public identities which could be used as their public keys (instead of mathematically generating public keys) to encrypt messages. Users could then use their private keys (corresponding to their public IDs) to decrypt messages. In this scheme, there is a “Private Key Generator” (PKG) which is responsible for generating private keys corresponding to the users’ public identities. This scheme has many advantages. For instance, users do not need to have many different public IDs associated to different CAs. Instead, they have a single public ID (e.g. their email addresses) and different PKGs, which issue different private keys corresponding to their public IDs (based on the system parameters). Using this method, users do not need to verify the recipient’s certificate: the recipients will be authenticated

at the time they request private keys from the PKG. The PKG generates users' private keys using the master key which is available only for PKG. Network public information is available for all users but only PKG can compute private keys, since it is the only entity that has the master key.

Since in structured P2P networks, each node is identified by a unique ID in the network, and all the messages within the network are routed based on these IDs, IBE seems a perfect choice for P2P networks. That is, these IDs could be used as nodes' public IDs in IBE scheme which also could be utilized in the message routing process of P2P networks.

Since IBE is a computationally expensive method, proposed methods for utilizing IBE in P2P networks are not applicable for message routing and are only suitable for the download phase. The reason for this, as previously mentioned, is that in the search process, a search query might be passed to tens of users until reaches its destination. Therefore, this work strives to effectively apply the IBE in message routing in P2P networks to secure them against outside attackers who are able to alter/modify the messages while being transmitted to their destinations.

1.2 Scope and Goal

As was mentioned earlier, P2P networks could be categorized into two main categories: structured and unstructured. The main focus of this work pertains to the latter, structured P2P networks. A P2P storage utility by Rowstron and Druschel [5] is an example of these networks.

There are many different topologies for structured P2P networks. Notwithstanding, according to Tanenbaum and Van Steen [1], since Chord topology which is proposed by

Stoica et al. [2] represents most of these topologies, this work has chosen Chord as the structured P2P topology to work with. It is expected that the methodology proposed in this work could also be applied to other structured P2P topologies (albeit with some modifications) and improve their performance.

This work assumes that the nodes in the network are honest: they do not malfunction in the network. Therefore, this work provides a strategy to secure the message routing process against the outsider attacker who is able to overhear the routing messages and alter/modify them.

The primary objective of this thesis is to design and develop a strategy for message routing in Chord (generally in structured P2P networks): making the nodes able to securely transfer search queries to each other (against outside attackers who are able to alter/modify the messages while being transmitted to their destinations) without losing the performance for the network. That is, the delay that users will encounter by applying this strategy is comparable to the situations in which there are no security methods applied. In addition, this strategy does not impose extra communication overhead for the network. The focus of this work is on efficient secret key sharing among the nodes at the time of message routing. In order to encrypt/decrypt the messages (making them secure) in the message routing process, a secret key must be shared between sender and receiver. By sharing this secret key, fast and efficient symmetric-key algorithms (instead of PKC algorithms) could be used for encrypting/decrypting messages. Therefore, this secret sharing must also be done efficiently while queries are being routed among users in the network.

1.3 Contribution

As previously mentioned, the objective of this work is to design and implement a strategy which makes it possible to efficiently and securely forward search queries to the participating nodes in structured P2P networks in the case of outsider attackers. The contribution of this work is as follows:

- Providing an efficient key sharing strategy among nodes in structured P2P networks while searching for contents which:
 - has acceptable performance when searching for contents. The performance of message delivery delay (waiting time to receive the result for the search query) is close to the situations in which there is no secure message routing applied.
 - is making it possible for the search queries to be encrypted/decrypted using a symmetric key. If the search queries are encrypted while being transmitted, the outside attacker cannot reveal the content of the query.
 - is not imposing any extra network communication to the network. In order to share secret keys among the nodes, they do not participate in any kind of process in which they have to send (or receive) any extra messages (network packets).
 - is also applicable for memory-constrained environments.
 - is highly scalable. It performs well in environments with large amount of users and by increasing the number of users in the network, the practicality of this method is not affected.

In the following chapters, using the proposed methodology, the above mentioned contribution will be studied and evaluated.

1.4 Thesis Outline

The following chapters of this research are structured as follows. Chapter 2 studies the background and related works to this thesis. Chapter 3 describes the proposed methodology for structured P2P networks, specifically for Chord. Chapter 4 demonstrates the results of applying proposed methodology to these networks. In the last chapter, the summary of this work and the possible future work will be presented.

Chapter 2

Related Works

In this chapter, a preliminary discussion about cryptography will be presented. This discussion also covers the IBE scheme in more details, which is the main focus of this work. This discussion is then followed by the concept of P2P networks. The “Chord” protocol will also be discussed in details, since it is the structured P2P overlay used in this work. This chapter also covers the security issues with message routing in P2P networks. Lastly, some of the proposed methods for employing IBE in P2P networks are presented.

2.1 Cryptography

Information security deals with different objectives. In other words, to securely transfer data over an insecure channel, several objectives must be satisfied. According to Menezes et al. [6] the main objectives could be categorized as the following :

- Confidentiality: No one but those who are authorized must be able to reveal the data.

- Integrity: Ensuring that data is not modified or altered by unauthorized users.
- Authentication: Confirmation of the identity of a user (s/he is who s/he claims to be).

The study of mathematical methods to achieve the above-mentioned objectives is called cryptography [6]. Cryptography can be divided into two main categories: Symmetric-Key cryptography and Public-Key cryptography.

In symmetric-key cryptography, both the sender and the recipient use the same key to encrypt and decrypt. In other words, when someone (sender) encrypts data using a key k and sends it to another person (recipient), the recipient must use the same key k to decrypt the message. One of the fastest and most well-known symmetric algorithms is the “Advanced Encryption Standard” (AES) by Daemen and Rijmen [7]. This algorithm can be used in three different key sizes: 128, 192 and 256 bits. A longer key size implies a longer execution time and, consequently, higher security against brute-force attacks.

Symmetric-key cryptography has some disadvantages. For instance, this method needs a prior knowledge of the key for the parties involved in this process. In other words, parties have to share a key before having a secure communication and this seems to be a very challenging task in many situations. In addition, each pair of users needs to have a different shared key than others in the network. Therefore, for large networks, this issue becomes very challenging. Users may need to store and manage many keys for different users in the network, which could become a very challenging task. Disadvantages of symmetric-key cryptography led to the invention of public-key cryptography (also known as asymmetric cryptography).

The concept of public-key cryptography (PKC), initially introduced by Diffie and Hellman in 1976 [8]. In PKC, unlike the symmetric-key cryptography, the encryption

key and the decryption key are not the same (even though they are mathematically tied together). The sender would encrypt the message with one key (the recipient's public key) and the recipient would decrypt it using a different key which is his own private key. Therefore, in PKC, the users in the network do not need pre-shared keys in order to have a secure communication over an insecure channel. One of the most well-known algorithms in PKC was invented by Rivest, Shamir and Adleman in 1978 [9], referred to as RSA.

In PKC, public keys need to be certified by an entity which is called a Certificate Authority (CA). The reason for this is that users must be able to prove that they are in fact who they claim to be. In other words, when someone wants to encrypt a message using the recipient's public key, he must be sure that he is using the actual recipient's public key and not someone else's. If someone pretends to be the actual recipient and provides his own public key, then he would be able to learn the message which was initially intended for the real recipient. These certificates can expire or be revoked by the CA. As a result, certificate management, both revocation and reissuing, in PKC is a challenging task and creates undesired communication overhead. For example, if one certificate is renewed, all other users (who are in contact with this user) must contact the CA to have this certificate verified as well. Therefore, as the numbers of users grow, this process becomes less efficient and less practical.

It is also worth mentioning that by utilizing PKC, nodes may need to store a relatively large number of certificates, which could become problematic in space constrained environments.

In order to simplify certificate management in PKC, a new approach, IBE, was suggested in which users' identities could be used as their public keys.

In the next section, the IBE will be thoroughly discussed.

2.2 Identity-based Encryption

Shamir initially introduced the concept of IBE in 1984 [4]. He suggested that any arbitrary string could be used as a user's public key, such as the user's email address. For example, a message sent to 'bob@company.com' could be encrypted by this email address as his public key.

In IBE, there is a Private Key Generator (PKG) that is in charge of generating private keys associated with users' public identities. PKG generates the system parameters (public information) as well as the master key (master private key for the system). As mentioned, any arbitrary string could be used as a public key for encrypting a message. The recipient then has to contact the PKG in order to get the corresponding decryption key to the key which was used in the encryption process. In the previous example, when Bob receives a message, which is encrypted using 'bob@company.com', he would contact the PKG to get his private key corresponding to 'bob@company.com' [10]. Figure 2.1 demonstrates the general idea of IBE.

Even though IBE fits in the public key cryptography definition (the encryption and decryption keys are different), it will be referred to as the IBE (and not PKC) in this work to distinguish between schemes in which any chosen strings can be used as public keys.

Unlike PKC, public and private keys in IBE are based on the users' IDs and are not required to be generated concurrently. This idea has several advantages such as the fact that a message can be encrypted even before its corresponding private key is generated (offline encryption). In other words, since the private key can be generated at anytime, it can be generated even after the encryption is done. In addition, since public keys are based on users' IDs, users do not have to check for the validity of other users'

Figure 2.1: General diagram of IBE

certificates, which eliminates the issues related to the certificate management. After this idea proposed by Shamir, many IBE schemas were proposed such as the ones presented by Desmedt and Quisquater [11], Maurer and Yacobi [12], Tsujii and Itoh [13] and Tanaka [14]. However, Boneh and Franklin [10] in 2001 proposed the first fully functional IBE scheme which was based on “Elliptic Curve Cryptography” (ECC) and “Weil pairing”.

2.2.1 Applications for IBE

The fact that public key and private key are not generated at the same time in IBE provides some unique applications for this system. Authors of paper [10] have introduced some of these applications as follows:

Revocation of Public Keys

In PKC, expiration dates of certificates are used to determine whether a certificate is expired or not. Therefore, users have to contact CA before sending the message. However

in IBE , the key expiration can be done by adding a time stamp to a public key. For example, if Bob wants to send a message to Alice, he can use “alice@company.com || current-year” as her public key. Therefore, Alice’s private key automatically expires after a year. An important point here is that the sender does not have to check for the validity of the recipient’s key in this scheme. It is even possible to change the expiration duration as the sender desires.

Managing user credentials

Adding credentials to the public key of recipients will provide the functionality of credential management. For example, if Alice sends a message to Bob using “bob@company.com || Clearance =secret” as Bob’s public key, Bob can only decrypt the message if he has a secret clearance.

Delegation of Decryption Keys

Let us imagine Bob has his own company and uses his own servers and has his own IBE system and he is also his own PKG. Therefore, he generates his own IBE parameters and since Bob has the master key, he can decrypt anything (no one else has access to this master key). Alice encrypts emails using the subject line of her emails (she could select anything as the subject). Since Bob has the master key, he can decrypt any messages from Alice. Now let us imagine that Bob has several assistants in different tasks such as “Accountant” and “General Manager”. Bob can now give one private key to each of his assistants according to their tasks in the company. Assistants can decrypt only the messages which have their duties in the subject line of the email. As a result, Alice only uses one public key (Bob’s parameters) for encrypting any email messages. The email could be decrypted only by the corresponding assistant to the email subject.

2.2.2 Drawback

Like any other system, IBE has disadvantages as well. As mentioned earlier, the PKG generates the system parameters and has access to the master key. As a result, the PKG can decrypt any messages for any users in the network. This problem is called the “Key-escrow” problem. Therefore, PKG should be trusted by the network users. This issue is also known as “Trust” problem. One way of tackling this problem is the use of “Threshold Cryptography” [15]. Instead of storing all the necessary information about the master key on one PKG, one must store different parts of this information on different machines in the network (multiple PKGs). In this way, no one has all the necessary information for creating any desired decryption key. When users need the corresponding decryption keys to their public IDs, they contact all those PKGs (or m out of n of those PKGs) to have their decryption keys constructed.

In addition, the mathematics behind the IBE systems is computationally more expensive than their equivalents in symmetric key cryptography and PKC. However, using IBE only for sharing the secret key before starting the communication could make this less expensive.

2.2.3 Preliminaries

The following are concepts used in the pairing-based IBE algorithms.

Cryptographic Bilinear Maps

The IBE scheme depends solely on the bilinear mapping of two groups of points on a curve. According to Dutta [16], a cryptographic bilinear map function is defined as follows:

Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups of the same prime order q and $\mathbb{G}_1$ is an additive and $\mathbb{G}_2$ is a multiplicative group. In addition, let P be an arbitrary generator of $\mathbb{G}_1$. A mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is called a cryptographic map if (aP means P has been added to itself a times):

- $e(aP, bQ) = e(P, Q)^{ab}$ for every $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$. This property is also called *Bilinearity*.
- $e(P, P) \neq 1$. This property is known as *Non-Degeneracy*.
- There must be an efficient algorithm for computing $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$. This property is known as *Computable*.

Diffie-Hellman Problems

The security of IBE is based on the difficulty of the Bilinear Diffie-Hellman as follows:

Bilinear Diffie-Hellman (BDH):

Having $\mathbb{G}_1$ and $\mathbb{G}_2$ as two groups of prime order q , $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ as a bilinear map and P as the generator of $\mathbb{G}_1$, given $\langle P, aP, bP, cP \rangle$ for some $a, b, c \in \mathbb{Z}_q^*$, it is difficult to compute $e(P, P)^{abc} \in \mathbb{G}_2$ [10].

The scheme that we have used in this thesis is based on difficulty of Bilinear Diffie-Hellman problem.

MapToPoint

In order to convert an arbitrary string (the identity of users) to a point on the curve, a function called “MapToPoint” is introduced by [10]. This function performs as follows:

Let E be the elliptic curve, for example, $y^2 = x^3 + 1$ over $\mathbb{F}_p$ where $p = 2 \bmod 3$ and $p = lq - 1$ for some prime $q > 3$. Also, let H be a cryptographic hash function: $H : \{0, 1\}^* \rightarrow \mathbb{F}_p$. Given the arbitrary string ID , the MapToPoint will be as follows:

1. $y_0 = H(ID)$.
2. $x_0 = (y_0^2 - 1)^{1/3} = (y_0^2 - 1)^{(2p-1)/3} \in \mathbb{F}_p$
3. $Q = (x_0, y_0) \in E(\mathbb{F}_p)$.
4. $Q_{ID} = lQ$.
5. return Q_{ID} .

2.2.4 IBE Algorithms

The following are the four different algorithms for IBE defined in [10].

Setup

In this phase, the parameters for the system, also known as *params*, such as elliptic curve parameters, and *master-key* will be generated. These parameters also define the message and the ciphertext finite space, M and C accordingly. As was mentioned earlier, these parameters are public but the master-key will remain private for the PKG of the system:

Given $k \in \mathbb{Z}^*$ as the security parameter which indicates the number of bits for q which is the order of groups of points:

1. Generate a prime q and two groups of order q , $\mathbb{G}_1, \mathbb{G}_2$, and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and a random generator $P \in \mathbb{G}_1$.
2. Choose a random $s \in \mathbb{Z}_q^*$ and define $P_{pub} = sP$.

3. Choose two cryptographic hash function, H_1 and H_2 , where $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$ for some n .

Therefore, the message space will be $M = \{0, 1\}^n$ and the ciphertext space will be $C = \mathbb{G}_1^* \times \{0, 1\}^n$ and the system parameters will be $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2)$ and the master key will be $s \in \mathbb{Z}_q^*$.

Extract

This algorithm takes system parameters, master-key and an arbitrary $ID \in \{0, 1\}^*$. This ID is the public ID that is being used to encrypt a message. The output of this algorithm is the corresponding private key to this ID:

1. $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$
2. $d_{ID} = sQ_{ID}$
3. Return the private key d_{ID}

Encrypt

This algorithm takes ID and $m \in M$ as the plaintext and returns $c \in C$ as its corresponding ciphertext using public key ID :

1. $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$
2. $g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*$
3. Choose a random $r \in \mathbb{Z}_q^*$
4. $C = (rP, M \oplus H_2(g_{ID}^r))$
5. Return the ciphertext C .

Decrypt

This algorithm decrypts an input $c \in C$ using the private key d_{ID} and returns the plaintext $m \in M$:

1. Let $C = (U, V)$
2. $M = V \oplus H_2(\hat{e}(d_{ID}, U))$
3. Return the plaintext M

To verify that the decryption algorithm works, it is needed to show that the following statement is true:

$$M \oplus H_2(g_{ID}^r) \oplus H_2(\hat{e}(d_{ID}, rP)) = M$$

Since the same hash function is used, it is needed to show that $\hat{e}(d_{ID}, rP) = g_{ID}^r$:

$$\begin{aligned} \hat{e}(d_{ID}, rP) &= \hat{e}(sQ_{ID}, rP) \\ &= e(Q_{ID}, P)^{sr} \\ &= e(Q_{ID}, P_{pub})^r \\ &= g_{ID}^r \end{aligned}$$

Therefore, the consistency of the algorithm is proved.

As shown, IBE depends solely on the bilinear mapping of two groups of points. Any function that satisfies this bilinearity property could be utilized in IBE. “Weil Pairing” has been utilized in [10] as the pairing function. Full description of Weil pairing and its efficient algorithm could be found in [17] by Miller.

2.2.5 Identity-Based Non-Interactive Key Exchange

Pairing could be utilized in key exchange. Non-interactive key exchange could be implemented using pairing [18]. Suppose users A and B would like to agree on a secret key without any pre-communication. In order to do so, they can benefit from the Identity-based key exchange. That is, node A and B can compute the secret s as follows:

$$K_A = e(sQ_A, Q_B)$$

$$K_B = e(sQ_B, Q_A)$$

Where Q_A is a point on the curve corresponding to the public identity of node A (similar to MapToPoint in the above) and sQ_A is its private key, which is given by the PKG. Q_B and sQ_B are node B 's public ID (its corresponding point on the curve) and private key respectively. Lastly, s is the system master-key which is unknown for both A and B .

According to the bilinearity property of the pairing, the following demonstrates that both A and B share the same key:

$$K_A = e(sQ_A, Q_B) = e(Q_A, Q_B)^s$$

$$K_B = e(sQ_B, Q_A) = e(Q_B, Q_A)^s = e(Q_A, Q_B)^s = K_A$$

Therefore, pairing could be used for sharing a key between two users without any pre-communication. It is sufficient to know their private keys (which are given by the PKG of the system) and each other's public identity.

This work primarily utilizes the above-mentioned technique to secure the process of

message routing in structured P2P networks which will be examined thoroughly in the next chapter.

2.3 Peer-to-Peer Networks

File sharing has become very popular especially with the rapid growth of the Internet, both geographically and technologically. File sharing has been used for a long time through the traditional centralized approach: a server stores all of the files and all clients, using the server, can access the shared files.

This traditional approach has its own disadvantages. The problem of server being the bottleneck is one of the most important disadvantages of this architecture. As the number of users and shared files grow, requests for downloading files will increase and consequently, the system's performance will significantly decrease. On the other hand, the most significant advantage of such a system is its ease of implementation. The facile implementation of a client-server file sharing system could be the reason this architecture has been widely utilized.

Despite the advantages of the client-server architecture, its many disadvantages led researchers to explore other file sharing architectures. P2P is the new generation of network architectures which appears to be a good replacement for the traditional client-server architecture. According to Doyle [19], "P2P is a class of applications that take advantage of resources storage, cycles, content, human presence available at the edges of the Internet. Because accessing these decentralized resource means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must operate outside the DNS and have significant or total autonomy of central servers". In other words, in P2P networks, the functionality is pushed to the users, unlike client-

server, in which the functionality is all on the server (which is also the cause of the bottle neck problem). In P2P architecture, each user provides his/her own bandwidth as well as a computational device to the network. Therefore, by having more users in such a network, the network becomes more robust and reliable. Figure 2.2 compares P2P and client/server architectures.

Figure 2.2: *P2P architecture against client/server architecture*

According to Bo and Li [20], the lifetime cycle of a P2P network could be divided into 4 phases: *join*, *query*, *download* and *depart*. At first, a node must join the P2P network at which point it could get some information about the network (e.g. neighbours info) as well as publish information about the files it provides for the network. Then, the node could begin querying for the desired files to find them in the network. After finding the destination (a node which holds the searched-for file), it can directly connect to the destination and download the file (if the file is found). Eventually, the node may let others know of his departure in case he leaves the network. All these four phases are achieved solely by the nodes in the network. It is worth mentioning that when a node is downloading a file, it could leave the P2P network if it wishes. In other words, it will

not need to, for example, participate in routing other nodes' queries while downloading and it can leave the network. As was mentioned earlier, this work mainly focuses on the query part of P2P networks.

Based on the object (file) query algorithms in P2P networks, they can be categorized into three different categories [21]:

- **Centralized :**

In centralized P2P networks, objects are stored in different nodes across the network. However, all object indexes are stored in a centralized server. Nodes, at arrival time, contact this server and provide it with the objects they may want to share. They can also query for the objects they are trying to find in the network, which they can receive in the form of $\langle \text{object-key, destination-address} \rangle$ from the server [20]. They can then contact the destination node and download the object.

Centralized P2P networks are very easy to implement. However, they suffer from the single failure point. If the server shuts down, the whole network will stop working. Having multiple servers is recommended to avoid this single failure point. An example of centralized P2P network could be the "Napster" [22]. In Napster, nodes would login to a server and provide a list of the files they are providing for other nodes in the network. Also, they send their queries to search for the files they desire which could be directly downloaded from the actual destination. In Napster, nodes can also search based on the keywords (objects which contain two or more keywords).

- **Decentralized-Unstructured :**

In decentralized-unstructured P2P networks, similar to the centralized ones, objects

are stored in different nodes all over the network. However, the search queries are also distributed across the network. That is, there is no centralized node responsible for storing all object indexes in the network. In this type of network, the search queries are usually randomly passed to other nodes in the network (hop-by-hop). In the search process, the object may be found or the query may time out (time-out value can be defined in the algorithm).

Unstructured P2P networks do not suffer from a single failure point. However, the search could be very inefficient since it is randomly developed. An example of decentralized and unstructured P2P network could be the “Gnutella” [23]. In Gnutella, users must first connect to another Gnutella node in order to obtain a list of some present nodes in the network. When a node wants to search for an object, it sends the search query to its neighbors. Flooding is the most common algorithm used for this matter, that is, sending the queries to all the neighbors. Queries may continue until they reach their defined Time-To-Live (TTL). TTL is a system-defined variable which indicates the number of hops a query has taken since it was initially started. However, this flooding algorithm is not very scalable, as the number of nodes grows in the network, it may create a lot of undesired communication in the network. Therefore, it has been brought to the attention of many researchers such as Lv et al. [24], Cohen and Shenker [25] and *Kangasharju et al.* [26].

- **Decentralized-Structured :** In Decentralized-structured P2P networks, similar to the unstructured ones, both object and search queries are handled in a distributed manner. However, the search query (and the network topology) is constructed based on a deterministic process and is not randomized. The most common procedure

to achieve this is a “Distributed Hash Table” (DHT). In systems which use DHT, there usually exists a large identifier space (e.g. 128-bit). Objects (files) and nodes (users) in the system are assigned a random number from this space. The most important task of such systems would be to design a deterministic algorithm to uniquely map each object’s number (object key) to a node’s identifier. Therefore, when someone is looking for an object in the network (using its key), the network address of the node, whose identifier is mapped to that key, must be returned [1].

In structured P2P networks, the search queries are still carried out hop-by-hop and passed according to the logically created network overlay. There are many structured topologies such as “Tapestry” by Zhao et al. [27], “Pastry” explained by Rowstron et al. [28] and “Butterflies” that is discussed in [29] by Datar. An overview of the data querying for P2P networks could be found in [30] by Balakrishnan et al. According to [1], “Chord” topology represents many of the structured topologies. Therefore, this work mainly focuses on Chord as the structured topology for P2P networks. The following will thoroughly elaborate on the Chord topology.

2.3.1 Chord

Chord [2] is a structured and decentralized look-up mechanism for P2P networks. In Chord, there is a large identifier space from which the nodes are randomly assigned an ID. One way for a node to get a random ID from this space could be to calculate the hash value of its IP address. This hash value is then used as the node’s ID and the hash algorithm would define the length of this space. For example, if SHA-1 presented in FIPS PUB 180-1 [31] (a widely-used hash algorithm) is used, this length would be 160-bit long. Considering the large size of this space, such as 160-bit (2^{160} items), it is

nearly impossible for nodes to end up with the same id in the network. Objects could be assigned a key in the same manner. That is, their hash value (their names, keywords or so on) is calculated and assigned to them. Therefore, all nodes and objects in the network are assigned a unique value from the same identifier space.

Nodes form a logical ring in Chord (based on their IDs). Any object with key k is mapped to a node whose identifier is the smallest id that $id \geq k$. This node is called the successor of this key, $succ(k)$. Therefore, if a node in the network would like to search for the object with key k (using the function $LOOKUP(k)$), it eventually needs to get the network address associated with $succ(k)$. Figure 2.3 on page 27 demonstrates a sample of how the data objects are mapped to the actual nodes in the network (ring). As it shows, the identifier space is 4-bit long (16 items in total) and nodes with identifiers 1, 4, 6, 11 and 15 are the actual nodes in the network and the rest are the data objects which are mapped to these nodes. For example, node 11 is responsible for data objects with keys 7, 8, 9, 10 and 11. If a node searches for any of these IDs, it will get the network address of node 11 which is responsible for them [1].

When a node wants to join the network, it has to first obtain a random id from the identifier space. Then, it has to find the node in the network that is currently responsible for that id. This could be done, as was mentioned earlier, via the $LOOKUP(id)$ function. This function will be returning the network address of $succ(id)$. In order to call this function in the network, the newly joined node contacts an arbitrary node in the network and revokes that function. After finding the responsible node for this id, the new node would contact this node and its predecessor in the ring and insert itself between them in the ring. As it can be seen, this method needs the nodes to store the information about their predecessors as well. After the node is successfully inserted, all the data objects which are now associated with the recently joined node must be transferred to the new

Figure 2.3: *General ring topology in Chord [1]*

node (from former $\text{succ}(id)$).

For example, in figure 2.3 on page 27, if a new node with id 9 wants to join the network it first has to call the function *LOOKUP*(9). This function returns the network address of current *succ*(9) which is node 11. The new node will then contact nodes 11 and its predecessor, node 6, and insert itself between them in the network. Finally, the data objects 7, 8 and 9 will be transferred to the node 9 (if there is any data object). Also, nodes 6 and 11 will have to update their successor and predecessor respectively. Figure 2.4 on page 28 shows the new ring after the node 9 has joined the network.

When a node wants to leave the network, it would simply contact its successor and predecessor and transfer the data objects (if needed) to its successor.

As mentioned earlier, the most important and challenging part of DHT-based systems is designing a deterministic search algorithm. In Chord, one naive implementation of this

Figure 2.4: Network topology after insertion of a node

algorithm could be to have each node with the identifier n store the information about its predecessor, $pred(n)$, and its successor, $succ(n)$. In this way, if node n receives a search query for key k , it would simply forward this query to either its predecessor or successor respectively. In the case that k happens to fall between n and its predecessor, $pred(n) < k \leq n$, node n would return its own network address as the result of the search query, since it is responsible for the key k . It is clear that this method is not scalable, since as the number of nodes grows, the number of hops will grow linearly.

In Chord however, in order to make the search scalable and feasible, each node stores a table with the maximum of m rows (for m -bit identifier space) called “Finger Table”. Equation 2.1 outlines the i^{th} row of this table for a node with the identifier of n as follows:

$$FT[i] = succ(n + 2^{i-1}) \quad (2.1)$$

As it can be seen, the i^{th} row of node n 's finger table, refers to the first node with an id that has a distance of at least 2^{i-1} from n . These references can be seen as short-cuts to ids for the actual nodes in the network. Therefore, using this finger table, when node n receives a search query for key k , instead of forwarding it to its predecessor or its successor, it would forward it to a node with the identifier of q where equation 2.2 defines q as follows:

$$q = FT_n[j] \quad \text{where} \quad FT_n[j] \leq k < FT_n[j+1] \quad (2.2)$$

Figure 2.5 on page 30 demonstrates this process for a search query for key 5 initiated from node 15. That is, node 15 looks into its finger table and according to the third row, it forwards this query to node 3, since $FT_{15}[3] \leq 5 < FT_{15}[4]$. Node 3 forwards this query to node 4 according to its first line in its finger table ($FT_3[1] \leq 5 < FT_3[2]$). Lastly, node 4 forwards this query to node 6 which is the destination for this query.

As presented, in finger tables, the distance between each row grows exponentially. Therefore, one can show that at most $O(\log(N))$ steps are needed to resolve a search query where N is the number of actual nodes in the network [1].

Since nodes join and leave the network frequently, they need to keep their predecessors, successors and finger tables up to date. In Chord, this can be done in a background process and on a regular basis. They need to regularly send a query for every row in their finger tables to keep them up to date. They also have to regularly contact their predecessor and successor to check if they are still valid (and if needed, update them), since nodes may depart the system due to failure. Details for these maintenance tasks could be found in [2].

Figure 2.5: Chord: Search process for key 5 from node 15

2.3.2 Message routing in structured P2P networks

Message routing is very crucial in P2P networks since many anonymous nodes could connect to these networks from any location and without any prior knowledge of others. As mentioned, the functionality of the P2P networks depends on their nodes. These nodes could also prevent the correct delivery of messages in the network. They are also able to alter/modify the messages before forwarding them. The nodes can even refuse to forward the messages or instead forward them to wrong destinations. Any of these could prevent the network from functioning according to the users desire [3]. Even those who are not members of the network could prevent the messages from being correctly routed. For example, if one member of the network is sending a message to another member of the network, a third person could overhear and alter/modify the message while transferring

the message (e.g. internet level routing devices). As a result, the receiver node may not realize that the message has been altered and therefore may follow a wrong routing path.

Throughout this work, any attack which is performed by a node within the network is referred to as an “Insider” attack and if the attack is performed by anyone outside of the network, it is referred to as an “Outsider” attack. In this thesis, we have provided a solution to identify an outsider attack from altering/modifying messages while being routed in structured P2P networks.

In the following, two of the security attacks which could lead to a faulty message routing in structured P2P networks are discussed:

Sybil Attack :

According to Castro et al. [3], in this attack, the attacker could selectively choose a node ID in the network close to the object the attacker desires. Therefore, the attacker can drop or alter/modify the queries for that particular object. The attacker can even represent itself by using a large number of node IDs in the network in order to take over a section of the network. As soon as the attacker takes control of a section in the network, it can control all the messages that pass through that section of the network.

This attack is performed by insider attackers. Outsider attackers could place themselves in strategic internet routing paths in order to be in charge of particular objects of the network.

Man in the Middle Attack (MITM):

This kind of attack could be performed when an attacker is placed between two different nodes in the network. All messages between these two nodes are routed through the attacker. In this case, the attacker can also alter the messages while s/he is passing them (or even create new messages) [32]. This attack is performed by insiders attackers. Outsider attackers could perform this attack as well.

In this thesis, we provide a strategy to secure the message routing process in structured P2P networks using the IBE against outsider attackers. In the following section, some works in the area of securing P2P networks using the IBE will be studied.

2.4 Identity-based Encryption in P2P Networks

As mentioned in 2.2, IBE performs based on the users' identities. This identity could be any arbitrary string of the user's choice. For example, it could be the recipient's email address. In the same network, using the same system parameters, this ID should be unique for each person (e.g. an email address). Otherwise, messages sent to users with the same ID could be read by all of them. This property could become challenging in some networks, especially when nodes do not have any information about other nodes in the network. In other words, in order to have a secure communication, nodes should have this information about their recipients in advance. As a result, obtaining other nodes' unique identities, such as their email addresses, may not be possible for all networks.

However, as mentioned earlier, all nodes in Chord use other nodes' unique IDs to send and receive messages within the network, and all message routings are performed based on these unique IDs. In other words, in such networks, the mechanism for providing a unique ID for each node is already established and nodes are informed about other nodes' IDs. Therefore, nodes need the certainty that messages are sent/received from the expected ID and are not altered/modified or revealed during the routing process. Since nodes' unique IDs are the primary tools for message routing in structured P2P networks, they can also be used as the nodes' unique identities in the IBE system. Therefore, this characteristic of structured P2P networks perfectly suits the requirements for the IBE systems.

A new node ID assignment in structured P2P networks has been suggested by Butler

et al. [33]. According to [33], most structured P2P networks become insecure when they assign unique identifiers to nodes and objects: “current assignment schemes allow an adversary to carefully select user IDs and/or simultaneously obtain many pseudo-identities leading ultimately to an ability to disrupt the P2P system in very targeted (and dangerous) ways” (such as earlier-mentioned Sybil attack). Therefore, this paper proposes a novel scheme, which is based on IBE. This paper has proposed three different protocols that have better performances compared to PKC.

Authors of [33] suggest to assign an ID to each node when joining the network and provide their corresponding private keys for them. Three different protocols were introduced by them as follows:

1. Decentralized scheme
2. Centralized scheme
3. Hybrid of the above approaches

When a node joins the network, in order to obtain its private key, it is authenticated using a call back: “all responses to requests are transmitted through a server-initiated TCP connection” [33]. In the following, the above-mentioned protocols are discussed in more details.

- **Decentralized scheme:**

In this protocol, the joining node initially contacts a trusted party in order to be assigned an ID and to obtain its corresponding private key. This node will next contact the network’s server and provide its ID and the time stamp that are both signed by its private key. The server then provides a signed copy of its ID and time

stamp to the node. Consequently, the Sybil attack is thwarted because the server only sends the signature to the IP address provided and does not issue multiple IDs for the same IP address. In addition, since it is signed by the server, other nodes could use it as an authentication method.

- **Centralized:**

In this scheme, the server performs the trusted party function as well as its server role. While the overhead in the network will be reduced, yet the bottleneck problem could still occur.

- **Hybrid scheme:**

This scheme is very similar to the decentralized scheme. However, the trusted party is one of the nodes in the network which is assigned by the server (could be authenticated by the other nodes) and is able to provide secret keys for the other nodes.

The proposed method by [33] secures structured P2P against Sybil attack since nodes are not able to selectively choose any desired ID in the network. It secures the message routing against any outsider attack since nodes can verify that messages are sent from the actual senders and are not altered/modified while being transmitted. However, this method imposes an undesired message delivery delay to the network during searching time. In chapter 4, the proposed method provided in [33] will be evaluated and compared to the proposed method of this thesis. It will be also shown that this thesis will highly improve the performance of message delivery delay using IBE.

Secure communication using the IBE inside a distributed system (specifically Chord) is suggested by Stading [34]. This method uses the nodes' identifiers as their public IDs

in IBE. It also uses the threshold cryptography to avoid the trust problem. That is, the nodes partially generate the master key of the system. When a node wants to obtain its private key, it has to contact those nodes that partially have the master key and then construct its private key accordingly.

Since nodes participate in the process of key generation and distribution, [34] also employed a strategy called “Bootstrapping”. This strategy explains that there must be a minimum number of nodes present in the network (system parameter) to start the key generation phase. The reason for this is that if the participants nodes in key generation are very limited, it will be much simpler for an attacker to compromise the system (he has to compromise a lesser number of nodes). In addition, if the number of participants is large enough, the attacker is less likely to be able to represent most of the nodes in the network (to have enough information about master key).

In the proposed method by [33], nodes would use IBE to encrypt messages among them. The public key for this is derived from their identity. The most significant bits of their ID in the network (system parameter) are part of the key and are concatenated with the valid period of time for that message (message cannot be decrypted after this period of time). The result is also concatenated with the public key of the system (generated after enough participants are present in the network). The IBE public keys used in Scribe are the following:

$$\text{public key } K \parallel \text{MSB ID} \parallel \text{Valid Timestamp}$$

where $\parallel$ is the concatenation symbol.

The proposed method provided in [34] secures structured P2P networks against outsider attacks, but it does not address the issue of Sybil attack. Instead, this method provides a strategy for not having only one entity as the generator of the master key

(use of threshold cryptography). However, similar to [33], the proposed method by [34] imposes an undesired message delivery delay to the network. Their proposed method will be evaluated and compared to this thesis and the results will be demonstrated in chapter 4.

IBE is also used in P2P devices, such as IP phones. Use of IBE-based authentication is proposed by Nguyen [35]. It has been suggested that network equipments, such as IP phones or network devices, could benefit from IBE in order to authenticate themselves. At the time when these devices are deployed, their private keys and system parameter could be generated and deployed with them. Later, when these devices attempt to communicate, they can authenticate themselves using pairing algorithm and next use the Diffie-Hellman algorithm to share a secret key.

When node A wants to communicate with node B :

- A Chooses a random m and sends “Hello B, I’m A” along with g^m to B
- B sends a random R_B to node A
- A computes hash of (P_{AB}, R_B, g^m) where P_{AB} is the pairing of A ’s private key and B ’s public key. It sends this along with a random R_A
- B computes the same thing and checks it against what A sent (since B can also compute P_{AB} using pairing). If it is verified, B chooses a random n and then computes hash of (P_{AB}, R_A, g^n) and sends it to A .

If A also verifies it, the shared secret between A and B will be g^{mn} . This scheme would provide a forward secrecy as well. Forward secrecy is when for every single communication, an independent session key is used. If a user’s private key is compromised,

the previous conversations cannot be revealed since they were encrypted using randomly generated session keys, and not users' public keys.

Since the devices may not be from the same domain (company), they might run under different system parameters. Therefore, they may not be able to authenticate themselves using this method. To resolve this issue, [35] also proposed a method called "Chaining" authentication. That is, domains could have Trusted Authority Gateways (TAG) and TAGs across different domains could share the same system parameters. As a result, if a node, A , from one domain wants to communicate with another node, B , from another domain, it would simply contact its own TAG and authenticate itself for it. After it is authenticated, A 's TAG would act as a proxy and forwards A 's request to B 's TAG, constructing a chain of authentication all the way to node B .

The proposed method provided in [35] secures the communication between P2P devices against outsider attacks. However, according to their proposed method, nodes always have to first agree on a new key for each session of communication. This process requires four extra data exchanges between the two parties (four send or receive in total). This strategy is more applicable in the downloading phase which is required to be performed only one time per download. Applying this method on the message routing (in search) not only imposes a significant message delivery delay to the network (for each step a new key must be assigned), but also imposes an undesired communication overhead to the network (four extra data exchanges are needed between the nodes at each step). In the next chapter, it will be shown that our proposed methodology in this thesis does not impose an undesired communication overhead to the network for securing the message routing against outsider attacks. In chapter 4, it will also be demonstrated that the message delivery performance of our proposed methodology is very close to the situations in which no security method is applied for message routing in the network.

2.5 Summary

In this chapter, background information for this work was covered. It included discussions about cryptography and specifically Identity-based Encryption. Also, general information about P2P networks and some of their security issues regarding message routing was provided. Chord protocol was also studied in depth. Lastly, some works done in P2P networks using IBE were discussed. In the following chapter, the proposed methodology will be described.

Chapter 3

Methodology

In this chapter, the proposed methodology for employing IBE in structured P2P networks will be presented. Two protocols will be introduced and discussed in details.

3.1 Introduction

A node's unique ID plays a very important role in structured P2P networks. It is not an overstatement to say that almost every feature in structured P2P networks is implemented using these unique IDs. In order to connect to a network and communicate with other nodes, unique IDs are utilized. In some protocols, such as Chord, these IP addresses have also been used for other advanced features [2]. For example, in Chord, these IDs are used to efficiently search the network. The uniqueness of these IDs motivated researchers to employ them as security measures for these networks. In this work, the same approach has been taken in order to introduce a method of securing message routing in structured P2P networks. These IDs will be used as primary tools for establishing secure message routing in structured P2P networks.

In the following, two protocols will be introduced to be employed in structured P2P using IBE.

3.2 Proposed Methodology

In order to access a specific content in a P2P network, one must first search for that content through the network - through the collaboration of other nodes in the network. When the node that contains the desired content is found, they could transfer the content via any desired protocol. As it can be seen, this process has two steps: searching for the content and transferring the content. This work only focuses on the former and does not cover the latter. That is, the protocols which are proposed in this work would only focus on the searching algorithm.

This work focuses on message routing in structured P2P networks, specifically the “Chord” protocol mentioned in 2.3.1. Figure 3.1 on page 41 demonstrates a simple search procedure in the Chord protocol. As presented, node A is searching for an item which existed on node D in the network. Node A first sends its search message, m , to node B (assuming this search follows this path based on the Chord protocol). Node B then forwards this message to node C who forwards it to node D, which is the destination node and has the item. As presented, there is always a one-to-one communication in this search. Each message is being transferred from one node to only one other node at a time.

Having only one-to-one communication at each time was the motivation behind this work. Since only two parties are involved, the IBE non-interactive key exchange algorithm could be applied with no extra communication between users to generate a shared secret key. In other words, nodes can agree on secret keys without any communication.

Figure 3.1: Simple search procedure in Chord [2]

Whereas, if there were more than two parties involved in the communication, extra communication would be required to share a secret key among all the parties (even while using the IBE). Therefore, in each step in the search, the message (search query) could be encrypted/decrypted via the shared key between the nodes which is generated using the IBE non-interactive key exchange algorithm.

It is worth mentioning that for any types of networks, in which nodes are assigned unique IDs and are aware of other nodes' unique IDs, the proposed method of this thesis is applicable. The reason for this is that if nodes are aware of other nodes' unique IDs, they can simply use non-interactive key exchange algorithm in order to agree on the same key to be used for encryption/decryption process.

In the following, two protocols will be presented to secure message routing in structured P2P networks. In all the protocols below, the nodes are assumed to be assigned a unique ID in the network and also provided with the corresponding private key.

It is worth mentioning that both protocols in this chapter use AES algorithm to encrypt/decrypt the messages and do not use the IBE algorithms for encryption/decryption. They also use a secure hash algorithm to produce a fixed length output. The second protocols also employs a “Key-Table” which is a hash table to store the necessary data for the protocol. The structure of this hash table will be discussed later in this chapter. The notations used in this work are described in table 3.1.

Table 3.1: *Notations used in the methodology.*

Notation :	Definition
N :	Nodes in the network
ID_A :	Unique ID for node A .
Pri_{ID_A} :	Corresponding IBE private key to the unique ID_A of node A .
$e(P, Q)$:	Pairing of the two elements P and Q according to 2.2.5.
$K_{A,B}$:	Shared key between nodes A and B .
$E_{aes}(m, k)$:	Encryption of message m using key k by AES algorithm.
$D_{aes}(m, k)$:	Decryption of message m using key k by AES algorithm.
$H(m)$:	256-bit long hash value of message m .
$Key - Table$:	A hash table to store generated keys for nodes in the network.

3.2.1 Protocol 1

In this protocol, when a node wants to join the network, it first contacts the server and obtains its unique ID as well as its corresponding private key and system parameters (IBE parameters). When a node wants to communicate with other nodes in the network, it generates the shared key for those nodes. For instance, when node A wants to send a message to node B , it would generate the shared key $K_{A,B}$ by which it encrypts the message and then sends it. Note that the shared key K_{AB} is generated using IBE Non-Interactive Key Exchange method which was mentioned in 2.2.5. This key is generated based on the pairing of the sender's IBE private key and the recipient's public ID, which is a unique key for any two nodes in the network. As a result, there is no need for any pre-communication between these two nodes. Since these keys are generated based on the users' private keys, the desired privacy is achieved in the network.

As mentioned in 2.3.1, nodes store the IDs of nodes that will communicate with them in their successors/predecessors list and finger tables. Therefore, in Chord, when a node wants to send a message to another node in the network it already knows about the recipient ID. This knowledge renders it possible to establish a shared key for every message sent/received in the network without any previous communication. When node A tries to send a message m to node B , it first encrypts the message using the "Encrypt" function and then sends it to the destination. Algorithm 3.1 outlines this function's pseudocode.

On the other end, when node B receives a message from node A , it needs to first decrypt the message using the "Decrypt" function and then encrypt it with a new key (for the new recipient). Algorithm 3.2 on page 44 outlines the decrypt function pseudocode.

Algorithm 3.1 Protocol 1 - Encrypt

Require: Pri_{ID_A}, ID_B, m

1. //Mapping B's public identity to a point on the curve according to 2.2.3
 2. $Q = MapToPoint(ID_B)$
 3. //Pairing A's IBE private key with Q
 4. $t = e(Pri_{ID_A}, Q)$
 5. //Hashing to have a 256-bit value
 6. $K_{A,B} = H(t)$
 7. //Encrypting the message using AES algorithm
 8. $m' = E_{aes}(m, K_{A,B})$
 9. **return** m'
-

Algorithm 3.2 Protocol 1 - Decrypt

Require: Pri_{ID_B}, ID_A, m'

1. //Mapping A's public identity to a point on the curve according to 2.2.3
 2. $Q = MapToPoint(ID_A)$
 3. //Pairing B's IBE private key with Q
 4. $t = e(Pri_{ID_B}, Q)$
 5. //Hashing to have a 256-bit value
 6. $K_{B,A} = H(t)$
 7. //Decrypting the message using AES algorithm
 8. $m = D_{aes}(m', K_{B,a})$
 9. **return** m
-

Figure 3.2 illustrates this protocol in which each step is marked by its order. As presented, node A first prepares the message query m , and wants to send it to node B . It then encrypts the message using the key $K_{A,B}$ which yields to m' and then sends m' to node B . When node B receives the message m' , it first decrypts it using the key $K_{B,A}$, which is identical to $K_{A,b}$, and reveals the original message m . Node B next encrypts the message m using the key $K_{B,C}$ and generates m'' and sends it to node C . Nodes C , in a similar manner, reveals the original message m and then prepares the message m''' and sends it to node D . Lastly, when node D receives the message m''' , it decrypts it with the key $K_{D,C}$, which yields the original message that was sent initially from node A .

Figure 3.2: Protocol 1

Discussions

As above, sending and receiving in this protocol are the same as the original Chord protocol. In other words, there is no extra communication imposed on the network in order to have the message securely routed, unlike the proposed method provided in [35] which includes four extra data exchanges between each two nodes. It is worth mentioning that each node in the network needs to perform additional work in order to encrypt and decrypt the message. This will increase the latency of the message delivery over the network.

For every message in the network, nodes have to compute the shared key every time, even for the same previously contacted destinations. This is the primary cause for an

increase in message delivery latency. In the following protocol, this issue has been addressed and the protocol has been modified to improve on this issue.

3.2.2 Protocol 2

In this protocol, like the previous protocol, when a node wants to join the network, it first contacts the server and obtains its unique ID as well as its corresponding private key and system parameters (IBE parameters). However, in this protocol, nodes benefit from the “Key-Table”, a hash table to store nodes’ unique IDs along with their generated shared key which is stored in nodes’ memory. Figure 3.3 demonstrates the structure of this table in this protocol. When a node wants to communicate with other nodes in the network, it first searches for the key-table to retrieve the desired key. If the key does not exist, it implies that it has never had a previous communication with the recipient node and therefore, it needs to generate the shared key as with the previous protocol. Any newly generated key will be stored in the Key-Table for further use. For instance, when node A wants to send a message to node B , it first checks whether it has the key $K_{A,B}$ or not. If it does, it will use the same key, otherwise it would generate the shared key $K_{A,B}$ and then store it in the Key-Table. As a result, there is still no need for any pre-communication between these two nodes like the previous protocol. When node A tries to send a message m to node B , it first encrypts the message using the new “Encrypt” function and then sends it to the destination. Algorithm 3.3 on page 48 outlines this function’s pseudocode. The new changes from the previous protocol are shown in bold. As presented, lines 2 and 12 are the key points in this algorithm.

On the other side, similar to the previous protocol, when node B receives a message from node A , it needs to first decrypt the message using the “Decrypt” function and

Figure 3.3: *Key-Table structure in protocol 2*

then encrypt it with a new key. Algorithm 3.4 on page 49 outlines the decrypt function pseudo-code.

Discussions

This protocol does not suffer from the performance issue as protocol 1 does. The reason being that nodes do not have to generate the shared key for every single message they want to transmit. Only when a node is sending (or receiving) a message to (from) another node for the first time, it needs to generate the corresponding key for that node. Otherwise, it can benefit from the previously generated and stored key and save the key generation time. However, this protocol might result in occupying much space for each node, since nodes store every single key they generate. Even if a message is sent to a node only once, the corresponding key is also stored in the memory. As a result, this could become problematic for storage-constrained environments. Since this solution could be utilized in any structured P2P network or, in general, any environment in which nodes have unique IDs and they are aware of other nodes' IDs, this issue must also be addressed. The following strategy which is a variant of protocol 2 addresses this problem.

Algorithm 3.3 Protocol 2 - Encrypt

Require: Pri_{ID_A}, ID_B, m

1. //Checking if the key already exists
 2. **if** ID_B **EXISTS IN KEY-TABLE** **then**
 3. $K_{A,B} = \text{KEY-TABLE}[ID_B]$
 4. **else**
 5. //Mapping B's public identity to a point on the curve according to 2.2.3
 6. $Q = \text{MapToPoint}(ID_B)$
 7. //Pairing A's IBE private key with Q
 8. $t = e(Pri_{ID_A}, Q)$
 9. // Hashing to have a 256-bit value
 10. $K_{A,B} = H(t)$
 11. //Storing the key and its corresponding ID for further use
 12. **INSERT** $(ID_B, K_{A,B})$ **INTO KEY-TABLE**
 13. **end if**
 14. // Encrypting the message using AES algorithm
 15. $m' = E_{aes}(m, K_{A,B})$
 16. **return** m'
-

3.2.3 Memory-Speed Trade-Off

This method is a variant of protocol 2. That is, nodes store the generated keys in their memory and before generating a key, they need to search for their key-table in order to retrieve it. If they cannot locate the key, they will have to generate it. However, in this method, the key-table is fixed in size and has limited capacity. Therefore, when nodes generate the keys, they need to first check whether or not there is enough memory left to store the key in the key-table (the capacity of the key-table is defined by the number of rows it can store and must be provided for the algorithm). If there is enough memory, they will just have to insert the new key into the Key-Table. However, if they do not have enough memory, they have to replace an existing key in the key-table with the newly generated key. An important point here is that the keys that are frequently used should have a longer life in the Key-Table as they improve the performance of the

Algorithm 3.4 Protocol 2 - Decrypt**Require:** Pri_{ID_B} , ID_A , m'

1. //Checking if the key already exists
2. **if** ID_A **EXISTS IN KEY-TABLE** **then**
3. $K_{B,A} = \text{KEY-TABLE}[ID_A]$
4. **else**
5. //Mapping A's public identity to a point on the curve according to 2.2.3
6. $Q = \text{MapToPoint}(ID_A)$
7. //Pairing B's IBE private key with Q
8. $t = e(Pri_{ID_B}, Q)$
9. // Hashing to have a 256-bit value
10. $K_{B,A} = H(t)$
11. //Storing the key and its corresponding ID for further use
12. **INSERT** $(ID_A, K_{B,A})$ **INTO KEY-TABLE**
13. **end if**
14. // Decrypting the message using AES algorithm
15. $m = E_{aes}(m', K_{B,A})$
16. **return** m

network. To clarify this point, it should be stated that if frequently used keys remain for a longer amount of time in the Key-Table, there remains a lesser amount of new keys to be generated. As a result, there will be a lesser amount of delay in the message routing process in the network. In order to achieve this, the following changes are made to the second protocol. A “counter” is stored along with any generated key in the Key-Table. Figure 3.4 on page 50 shows the structure of the key-table in this method. Every time a key is fetched from the table (to send/receive a message), this counter will be incremented. This counter initially is set to 1, as any newly generated key is used at least once, otherwise, the keys would not have been generated. As long as the Key-Table is not full, newly generated keys, along with their counters, will be inserted into the table. However, in case the maximum capacity of the table has been reached, the key with the least amount of counter will be replaced with the new key (and the counter will again be reset to 1). Therefore, the highly used keys last longer in the table. These keys could

belong to the nodes at the critical routing points in the network or the nodes which have the most popular contents among other nodes in the network. Algorithm 3.5 on page 51 shows the changes made in the encrypt function pseudo-code of protocol 2 for this method. The main changes are in bold.

Figure 3.4: *Key-Table structure in memory-speed trade off method for protocol 2*

On the other side, the receiver node would have a similar procedure in order to first decrypt the message and then encrypt it for the new recipient. Algorithm 3.6 on page 52 shows the decrypt function pseudo-code for this method.

In this method, nodes can have any memory size that is suitable for them. In other words, one node could store up to, for example, 10 keys at a time, whereas another node stores up to, for example, 25 keys. However, the more space the nodes dedicate to the key table, the better the network will perform. Ultimately, if the nodes have no constraint in terms of space, this protocol performs similar to the initial form of protocol 2.

3.3 Analysis

In the above mentioned protocols, use of the IBE method has been suggested. By employing this method, certificate management issue with PKC no longer exists. Depending

Algorithm 3.5 Memory-speed trade off method for protocol 2 - Encrypt**Require:** Pri_{ID_A} , ID_B , m

1. //Checking if the key already exists
2. **if** ID_B EXISTS IN KEY-TABLE **then**
3. $K_{A,B} = \text{KEY-TABLE}[ID_B][\text{Key}]$
4. // Increasing the corresponding counter
5. **Increment** **KEY-TABLE** $[ID_B][\text{Counter}]$
6. **else**
7. // Mapping B's public identity to a point on the curve according to 2.2.3
8. $Q = \text{MapToPoint}(ID_B)$
9. // Pairing A's IBE private key with Q
10. $t = e(Pri_{ID_A}, Q)$
11. // Hashing to have a 256-bit value
12. $K_{A,B} = H(t)$
13. //Checking the storage space in order to store the key
14. **if** **KEY-TABLE IS FULL** **then**
15. // Removing the least frequently used key from the storage
16. **RemoveItemWithLowestCounterValue**(**Key-Table**)
17. // Inserting the new key
18. **INSERT** ($ID_B, K_{A,B}$) **INTO** **KEY-TABLE**
19. **KEY-TABLE** $[ID_B][\text{Counter}] = 1$
20. **else**
21. **INSERT** ($ID_B, K_{A,B}$) **INTO** **KEY-TABLE**
22. **KEY-TABLE** $[ID_B][\text{Counter}] = 1$
23. **end if**
24. **end if**
25. // Encrypting the message using AES algorithm
26. $m' = E_{aes}(m, K_{A,B})$
27. **return** m'

on the environment, any of the above protocols could be used.

It is worth mentioning that the IBE method is a computationally expensive algorithm (1000 times slower than PKC [33]). Therefore, this could potentially render this approach inefficient. However, what makes this approach efficient in this work is that it partially uses the IBE algorithm. In other words, the proposed method does not encrypt or decrypt any messages explicitly using the IBE algorithms. Unlike the proposed methods provided

Algorithm 3.6 Memory-speed trade off method for protocol 2 - Decrypt

Require: Pri_{ID_B}, ID_A, m'

1. //Checking if the key already exists
 2. **if** ID_A EXISTS IN KEY-TABLE **then**
 3. $K_{B,A} = \text{KEY-TABLE}[ID_A]$
 4. // Increasing the corresponding counter
 5. **Increment** $\text{KEY-TABLE}[ID_A][\text{Counter}]$
 6. **else**
 7. // Mapping A's public identity to a point on the curve according to 2.2.3
 8. $Q = \text{MapToPoint}(ID_A)$
 9. // Pairing B's IBE private key with Q
 10. $t = e(Pri_{ID_B}, Q)$
 11. // Hashing to have a 256-bit value
 12. $K_{B,A} = H(t)$
 13. //Checking the storage space in order to store the key
 14. **if** KEY-TABLE IS FULL **then**
 15. // Removing the least frequently used key from the storage
 16. **RemoveItemWithLowestCounterValue**(Key-Table)
 17. // Inserting the new key
 18. INSERT $(ID_B, K_{B,A})$ INTO KEY-TABLE
 19. $\text{KEY-TABLE}[ID_A][\text{Counter}] = 1$
 20. **else**
 21. INSERT $(ID_B, K_{B,A})$ INTO KEY-TABLE
 22. $\text{KEY-TABLE}[ID_A][\text{Counter}] = 1$
 23. **end if**
 24. **end if**
 25. // Decrypting the message using AES algorithm
 26. $m = E_{aes}(m', K_{B,A})$
 27. **return** m
-

in [33] and [34] which encrypt and decrypt all the message using the IBE algorithm discussed in 2.2.4, this work only uses the IBE pairing algorithm to generate the shared key. The message will then be encrypted/decrypted using the AES algorithm.

To clarify, encrypting all the messages using only IBE and sending them requires the following steps:

1. Generating a random AES key

2. Encrypting the message using the key generated in the previous step
3. Encrypting AES key using the IBE algorithm

The protocols proposed in this chapter require the following steps:

1. Generating the AES key using IBE non-interactive key exchange
2. Encrypting/Decrypting the message using the key generated in the previous step

The second step in both methods are the same. Therefore, this step takes the same amount of time in both methods. In addition, step 1 in the first method does not excessively affect the performance of the system. Therefore, step 2 in the first method and step 1 in the second method are the ones that could make a difference in the performance of each method. However, as mentioned in 2.2, during the IBE encryption, a pairing algorithm is also taking place. Therefore, the second method could be seen as a subset of the first method and as a result, it is expected to perform better than the first one. In addition, the encryption is always done in the first method, since it uses a random AES key that should always be encrypted. However, in the two last protocols of the second method, the pairing is not necessarily done for every message, since the keys could be stored in the memory.

According to the above, it is expected for the proposed method in this work to perform significantly faster than a regular encryption done by the IBE algorithm.

In addition, unlike the method provided in [35] which requires four extra data exchanges between the nodes, the proposed method by this work, as was shown, does not require any extra communications between the nodes while routing the messages.

3.4 Summary

In this chapter, two protocols were introduced for securing the communication in the Chord protocol. In addition, it was shown that the proposed protocols are expected to perform better than securing the message routing by using the IBE-based encryption and decryption proposed in [33] and [34]. It was also shown that, unlike the method provided in [35], the proposed method does not impose any extra communications to the network while routing the messages.

In the following chapter, the proposed methodology will be evaluated and the results will be demonstrated. In addition, the proposed methodology will be compared to the situation in which the IBE-based encryption/decryption is used as a primary tool to secure the message routing in the network such as provided in [33] and [34].

Chapter 4

Evaluation and Results

In this chapter, the proposed methodology will be examined through different experiments. All proposed protocols in the previous chapter are implemented and compared and the results will be demonstrated and analyzed in the following sections.

4.1 Implementation

This work has been implemented in the “Peersim”, a well-known and widely used simulation framework for P2P networks [36]: “PeerSim has been developed with extreme scalability and support for dynamicity in mind”. Many P2P networks have been simulated using Peersim such as “Pastry” and “BitTorrent” [36]. Peersim has been implemented using the Java programming language. Therefore, this work is also implemented in Java. This work has also utilized the implementation of the Chord protocol, which is provided in [37], and has modified it for the purpose of this work.

For the implementation of cryptographic algorithms, Multi-precision Integer and Rational Arithmetic C/C++ Library (MIRACL) is used, which is mainly a tool for im-

plementing cryptographic systems [38]. All cryptographic algorithms have been implemented in Microsoft Visual Studio 2010 and have been imported to the Peersim framework as a dynamic link library (DLL). This DLL includes the functions for pairing, AES encryption/decryption and IBE algorithms.

It should also be mentioned that in this chapter, IBE-only method refers to the situation in which the messages are encrypted/decrypted using the IBE encryption/decryption algorithms explained in 2.2, which is similar to what [33] and [34] have proposed for securing communication in P2P networks.

4.2 Environment

All experiments in this chapter have been performed on a 2.00 Ghz AMD Turion 64-bit processor (dual core) with 3 gigabytes of RAM

Setup:

Table 4.1 describes the settings that have been applied in all the experiments of this chapter.

Table 4.1: *Settings used in this work's experiments.*

Number of nodes	2000
Minimum network delay	50 milliseconds
Maximum network delay	400 milliseconds
Chord ID length	128 bits
Number of randomly-generated messages	10,000
Elliptic curve used in the IBE algorithm	$y^2 = x^3 + 1 \pmod{p}$
p	512-bit random prime: $p = 2 \pmod{3}$

4.3 Experiments

For evaluating the proposed methodology, different experiments have been performed as follows. In each experiment, a network consisting of 2000 nodes as well as 10,000 random messages have been generated. These messages are then sent from/to randomly chosen nodes in the network. The messages have been categorized based on the number of hops they take to arrive at the destination. Since increase in the number of hops indicates the increase in the number of nodes in the network, these categories show the effects of scalability on the performance of each experiment. The average time of message deliveries in each category have been calculated and shown accordingly.

It is worthwhile to mention again that messages in the experiments are search queries initiated from the nodes and they are not file contents being exchanged.

The results for all the experiments will be demonstrated in this section. A thorough discussion and evaluation of all these experiments will be provided in section 4.4.

4.3.1 Experiment 1: No Security

In order to accurately evaluate the proposed methods, these methods also have to be compared to a communication scenario in which there are no security methods applied in the network. Messages are sent to the recipients without being encrypted in any ways. Figure 4.1 displays the results of this experiment. As this figure indicates, the minimum average message delivery time belongs to the category of messages with one hop. That is, a minimum of 226 milliseconds. The maximum average delivery time belongs to the maximum number of hops which is 11 hops for which the delivery time is 2433 milliseconds. It also shows the linear relation between the number of hops and message delivery time in the network.

Figure 4.1: *Message delivery time with no security methods.*

4.3.2 Experiment 2: IBE

In this experiment, the use of IBE algorithms for securing message routing in the network has been tested, similar to the IBE encryption/decryption algorithms which are used in [33] and [34] for securing the communication in the network. Similar to the previous experiment, figure 4.2 shows the results for this experiment. It follows the same trend as the previous experiment: the minimum average delivery time belongs to the minimum hop number and the maximum average delivery time belongs to the maximum hop number. However, in this experiment, the average delivery time has been significantly increased. The minimum average delivery time of 513 milliseconds belongs to messages with the minimum hop number, and the maximum average delivery time of 5429 milliseconds belong to messages with the maximum hop number. The linear relation between the

number of hops and message delivery time, as is shown, still remains.

Figure 4.2: *Message delivery time using IBE-based algorithms.*

4.3.3 Experiment 3: Protocol 1

In this experiment, protocol 1 has been tested. The shared key, according to protocol 1 in 3.2.1, is calculated with which the messages are encrypted/decrypted. Figure 4.3 shows the results for this experiment. This experiment also follows the same trend as the others. However, the delivery time has been further reduced. That is, the minimum average delivery time of 356 milliseconds belongs to messages with the minimum hop number, and the maximum average delivery time of 3835 milliseconds belongs to messages with

the maximum hop number. This protocol also keeps the linear relation between the number of hops and message delivery time.

Figure 4.3: *Message delivery time using protocol 1.*

4.3.4 Experiment 4: Protocol 2

In this experiment, protocol 2 has been tested. The shared key, according to protocol 2 in 3.2.2, is calculated with which the messages are encrypted/decrypted. The keys are also stored in the memory for further use. Figure 4.4 presents the results of this experiment. This experiment has the least average message delivery time compared to the previous two experiments, while still following the same trend. That is, the minimum average delivery time (i.e. 283 milliseconds) belongs to messages with the minimum hop number, and the maximum average delivery time (i.e. 2724 milliseconds) belongs to messages with the maximum hop number.

Similar to protocol 1, this protocol also maintains the linear relation between the

number of hops and message delivery time.

Figure 4.4: *Message delivery time using protocol 2.*

4.4 Discussion

As the results indicate, the IBE method has a significant impact on the network performance. As it can be seen, in every hop, the average delivery time has been increased dramatically. Figure 4.5 and 4.6 demonstrate this effect. As they indicate, this time has been increased by at least an average of 118 percent in every hop.

Figure 4.5: Comparison of average message delivery time between IBE-only and no security methods.

Figure 4.6: Comparison of average message delivery time between IBE-only method and no security method for each hop.

This phenomenon was expected to occur since, as mentioned earlier in this work, the IBE method is very computationally expensive. However, this impact could be reduced by partially using the IBE method, which was the main idea behind protocol 1 mentioned in section 3.2.1 (page 43). Protocol 1 reduced the average message delivery time, as it was expected and the results demonstrated. Figure 4.7 and 4.8 demonstrate that protocol 1 has reduced this time by at least an average of 29% for all of the hops.

Figure 4.7: Comparison of average message delivery time between IBE-only method and protocol 1.

Figure 4.8: Comparison of message delivery time between IBE-only method and protocol 1 for each hop.

However, this time reduction could be improved even more for environments in which the nodes are not space constrained and can store the generated keys. Figure 4.9 and 4.10 demonstrate that applying protocol 2 can reduce message delivery time in protocol 1 by at least an average of 21% for all of the hop numbers, which is a significant improvement.

Figure 4.9: Comparison of message delivery time between protocol 1 and protocol 2.

Figure 4.10: Comparison of message delivery time between protocol 1 and protocol 2 for each hop.

Depending on the amount of space dedicated for the nodes, Memory-Speed trade off method of protocol 2 performs between these two protocols. Clearly, the more space nodes dedicate to the keys, the closer the message delivery times will be to the initial form of protocol 2. Figure 4.11 demonstrates this effect. This figure compares the average message delivery time of all hops. As it can be seen, by increasing the number of keys stored in key table, the average message delivery time becomes closer to the protocol 2, which has no limit on the key table size. As this figure shows, by storing the maximum of 23 keys in the key table, almost the same performance as protocol 2 could be achieved for the speed-memory method. Figure 4.12 shows that while assigning the maximum stored number of keys to 23, the average number of keys stored by the nodes across the network is 14. Therefore, by having an average of 14 keys stored per node in the network, it is possible to achieve almost the same average message delivery time as what protocol 2 offers. It can be concluded that by storing even a small number of keys, a significant

message delivery time performance could be achieved.

Figure 4.11: *Effect of Key-Table size on average message delivery time in protocol 2.*

Figure 4.12: *Average number of keys stored in Memory-Speed method of protocol 2.*

In protocol 2, the network performance also depends on the number of messages exchanged among the nodes in the network. In other words, since nodes store the generated keys for their communications, the network performance for message delivery time improves as nodes become more likely to use previously generated keys for message routing. Figure 4.13 demonstrates this phenomenon. The average delivery time for the randomly generated messages improves towards the end of the experiment. As it can be seen, the first and second 1000 messages have significant difference compared to the rest of the messages and are closer to the output of protocol 1. The reason for this difference is that since there are not too many keys cached (previously generated and stored) in the beginning of the simulation (for each individual node), nodes are more likely to generate new keys. As new messages are generated, the number of stored keys increases and, as a result, the need for generating new keys decreases, which also results in improved performance (less message delivery time). Lastly, the least amount of message delivery time belongs to the last 1000 generated messages. This also verifies the above statement that the network performance depends on the number of messages which are sent/received in the network: as the number of messages sent/received in the network grows, the message delivery time decreases.

Figure 4.13: Comparison of average message delivery time when increasing the number of messages in protocol 2.

On a final note, figure 4.14 and 4.15 show the comparison among all previously mentioned experiments.

Figure 4.14: Comparison of message delivery time in all methods.

Figure 4.15: Comparison of message delivery time in all methods for each hop.

As demonstrated, protocol 2 performs similar to the situation in which no security methods are applied. Also, both protocols that were introduced in this work perform better than the IBE-only method that has been proposed by others such as [33] and [34]. Even in the case of the lowest performance, which is the protocol 1, it has a significant improvement over the IBE-only method. Table 4.2 provides a summary of all the above-mentioned methods and their comparison to the situation in which there is no security strategy applied.

Table 4.2: *Comparison of message delivery time performance of all methods against no security method.*

# Hops	No Security	Protocol 2	Additional Overhead	Protocol 1	Additional Overhead	IBE-Only in [33, 34]	Additional Overhead
1	226	283	25%	356	58%	513	127%
2	447	548	23%	698	56%	1005	125%
3	682	800	17%	1042	53%	1484	118%
4	906	1054	16%	1396	54%	1991	120%
5	1126	1302	16%	1738	54%	2488	121%
6	1357	1541	14%	2090	54%	2976	119%
7	1574	1793	14%	2433	55%	3465	120%
8	1790	2021	13%	2777	55%	3946	120%
9	2007	2237	11%	3119	55%	4438	121%
10	2224	2489	12%	3481	57%	4957	123%
11	2433	2724	12%	3835	58%	5429	123%
Average	1343	1527	14%	2088	55%	2972	121%

As presented, protocol 2 only increases the message routing time by an overall 14%, which has the best performance of all the methods. The IBE-only method has the least performance of all, in which the performance is decreased by an overall 121 percent.

4.5 Summary

In this chapter, various experiments on different methods for securing the message routing in the Chord protocol have been performed and the results were then compared. As demonstrated, protocol 2 has the highest and the IBE-only method has the lowest

performance of all the methods. Protocol 2 performs with an average of only 14 percent difference in message delivery time compared to the situation in which no security methods are applied. Therefore, this protocol can be a promising method in order to make message routing secure within the Chord protocol, in particular, and structured P2P networks, in general.

Chapter 5

Conclusions and Future Work

In this chapter, the summary of this thesis as well as possible future work are presented.

5.1 Conclusions

In this work, P2P networks and some of their security vulnerabilities regarding message routing were studied. Also, the conventional methods for tackling these vulnerabilities were discussed. A recently proposed method, the IBE, was also studied. As it was shown, the current problems with the conventional methods led researchers to consider the IBE method. These problems include the requirement to have a certificate along with each user in the network. As a result, nodes would have to verify these certificates in order to communicate with each other. As discussed, this process could become problematic for P2P networks in which nodes continuously join and leave the network. It was also shown that even though the use of IBE-based encryption/decryption could tackle this issue, it would impose a high performance loss to the network, since it is a very computationally expensive method. Therefore, two protocols were proposed to avoid this critical issue of

performance loss.

The proposed protocols benefit from the fact that nodes in structured P2P networks are assigned unique IDs and in order to route the search queries, they must be aware of the other nodes' IDs (those necessary for message routing). Therefore, these IDs could be utilized as arbitrary identities in the IBE method.

We proposed that instead of encrypting/decrypting the queries (or even symmetric shared keys) using IBE algorithm, a symmetric key could be generated using pairing algorithms in the IBE by which the queries could be encrypted/decrypted. It was shown that in order to generate this key, nodes do not need to communicate with each other and this key could be generated offline.

Two protocols were proposed by which the nodes could share secret keys in order to encrypt/decrypt the queries in the search phase of P2P networks. Protocol 2 also employs a table called "Key-Table" in order to store the generated shared keys in the memory for the future use. In this way, the keys which were previously generated can be used without being re-generated and, as a result, the search process could face a lesser amount of delay.

By applying the proposed methodology, the performance of message delivery time was significantly improved. Protocol 1, that had the least desirable performance among the proposed methods, improved the performance of message routing using the current IBE-based methods, as proposed in [33] and [34], by an average of 30 percent. Protocol 2 improves protocol 1's performance by an average of 25 percent. It was shown that protocol 2 imposes only an average of 14 percent message delivery performance loss to the situations in which no security methods are applied. Therefore, the proposed method does not highly affect the performance of the network when searching for the contents. In other words, the average message delivery time was only increased by an average of 14

percent compared to situations in which no security methods are applied. As a result, this is a very promising method for securing the search process in structured P2P networks.

In addition, memory-speed trade off method which is a variant of protocol 2 was introduced which is suitable for memory-constrained environments. This method, depending on the environment, could improve the performance within the boundaries of protocol 1 and protocol 2. This method stores a limited number of shared keys in the memory (rather than storing every single generated key). It was shown that by storing an average of only 14 keys per node in the network, almost the same performance as what protocol 2 offers could be achieved.

It was also shown that the proposed methodology does not impose any extra communication overhead among nodes to share a secret key for encrypting and decrypting the search queries, as opposed to the proposed method provided in [35] which requires extra communication between nodes in the network.

The proposed methodology was also proven to be highly scalable. When the number of hops in the process of message routing increases (as a result of the network growth), the proposed methodology does not lose the performance. This is because by increasing the number of nodes in the message routing process, the chance of having stored shared keys among these nodes also increases which leads to better performance.

5.2 Future Work

For future work, the following plans could be considered. First, the proposed protocols can be examined for other types of P2P networks, and not only structured ones. That is, even though nodes in unstructured P2P network do not need to have unique IDs across the network, they could implement an strategy to obtain one and then join the network.

Therefore, they can utilize these obtained IDs later on to secure the search process (e.g. by applying the proposed protocols in this thesis).

The implication of these protocols on current P2P networks such as “Skype” [39], can be other directions for future work. Skype does not only use the P2P overlay for searching for contents in the network, it also uses the overlay network for transferring contents, such as audio and video. Therefore, applying the proposed methods in this thesis could improve the quality of this service since proposed protocols significantly improve the message delivery time performance which is very important in live communication such as audio and video.

Also, in memory-speed trade off method of protocol 2, examining other caching methods such as “Last Recently Used” and etc. to find the most frequently used keys for each node are worthy of doing research which could lead to decreased amount of memory use for storing the keys, yet achieve the same performance as protocol 2. Additionally, since nodes in P2P network tend to join and leave the network repeatedly, it is worthwhile to apply expiry time for elements in the key-table. There could be cases that the elements in the table have a high number of counters but the corresponding nodes to them have already left the network. Therefore, an expiry time according to the nature of the network could be defined and its associated performance could be examined.

This thesis mainly focused on the search phase in P2P networks and the download phase was not looked at in details. Therefore, incorporating proposed methodology with the download phase in P2P network is a highly recommended task. That is, after the search phase is successfully performed in P2P networks, the search initiator and the node that has the content would start the download phase which means transferring the content. Since the search query starts and ends with these two nodes, it is possible to construct a secret key between these two nodes as the search query progresses to the

destination (in order to be used at the downloading phase). Otherwise, these nodes would have to engage in extra communication in order to agree on a key so as to exchange the content.

References

- [1] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*. Pearson Prentice Hall, October 2006. ISBN: 0-13-239227-5.
- [2] I. Stoica, R. Morris, D. Karger, F. M. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, vol. 31, (New York, NY, USA), pp. 149–160, ACM, October 2001.
- [3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, “Secure routing for structured peer-to-peer overlay networks,” in *Proceedings of the 5th symposium on Operating systems design and implementation*, OSDI ’02, (New York, NY, USA), pp. 299–314, ACM, 2002.
- [4] A. Shamir, “Identity-based cryptosystems and signature schemes,” in *Proceedings of CRYPTO 84 on Advances in cryptology*, (New York, NY, USA), pp. 47–53, Springer-Verlag LNCS 196, 1984.
- [5] A. Rowstron and P. Druschel, “Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility,” in *Proceedings of the eighteenth ACM*

- symposium on Operating systems principles*, SOSP '01, (New York, NY, USA), pp. 188–201, ACM, 2001.
- [6] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2001. ISBN: 978-0849385230.
- [7] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Berlin, Heidelberg, New York: Springer Verlag, 2002.
- [8] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, pp. 644 – 654, Nov. 1976.
- [9] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, pp. 120–126, February 1978.
- [10] D. Boneh and M. K. Franklin, “Identity-Based Encryption from the Weil Pairing,” in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 213–229, Springer-Verlag LCNS 2139, 2001.
- [11] Y. Desmedt and J.-J. Quisquater, “Public-key systems based on the difficulty of tampering (Is there a difference between DES and RSA?),” in *Proceedings on Advances in cryptology—CRYPTO '86*, (London, UK), pp. 111–117, Springer-Verlag LNCS 263, 1987.
- [12] U. M. Maurer and Y. Yacobi, “Non-interactive public-key cryptography,” in *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'91, (Berlin, Heidelberg), pp. 498–507, Springer-Verlag LCNS 547, 1991.

-
- [13] S. Tsujii and T. Itoh, “An ID-based cryptosystem based on the discrete logarithm problem,” *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 467 – 473, May 1989.
 - [14] H. Tanaka, “A Realization Scheme for the Identity-Based Cryptosystem,” in *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO ’87, (London, UK), pp. 340–349, Springer-Verlag LCNS 293, 1988.
 - [15] P. S. Gemmell, “An introduction to threshold cryptography,” *CryptoBytes Technical Newsletter*, vol. 2, no. 3, pp. 7–12, 1997.
 - [16] R. Dutta, R. Barua, and P. Sarkar, “Pairing-Based Cryptographic Protocols: A Survey,” in *Cryptology ePrint Archive, Report 2004/64*, 2004.
 - [17] V. S. Miller, “The Weil Pairing, and Its Efficient Calculation,” *Journal Of Cryptology*, vol. 17, pp. 235–261, September 2004.
 - [18] R. Sakai, K. Ohgishi, and M. Kasahara, “Cryptosystems based on pairing,” in *Symposium on Cryptography and Information Security (SCIS2000)*, (Okinawa, Japan), pp. 26–28, January 2000.
 - [19] J. F. Doyle, “Peer-to-peer: harnessing the power of disruptive technologies,” *Ubiquity*, vol. 2001, May 2001.
 - [20] C. W. Bo and B. Li, “Peer-to-Peer Overlay Networks: A Survey,” tech. rep., Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, 2003.

-
- [21] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, “Search and replication in unstructured peer-to-peer networks,” in *Proceedings of the 16th international conference on Supercomputing*, (New York, NY, USA), pp. 84–95, ACM, 2002.
- [22] “Napster.” <http://www.napster.com/>, Last visited on January 2011.
- [23] “Gnutella.” <http://rfc-gnutella.sourceforge.net>, Last visited on January 2011.
- [24] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, “Search and replication in unstructured peer-to-peer networks,” in *Proceedings of the 16th International Conference on Supercomputing*, (New York, NY, USA), pp. 84–95, ACM, 2002.
- [25] E. Cohen and S. Shenker, “Replication strategies in unstructured peer-to-peer networks,” in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM ’02, (New York, NY, USA), pp. 177–190, ACM, 2002.
- [26] J. Kangasharju, K. Ross, and D. Turner, “Optimizing File Availability in Peer-to-Peer Content Distribution,” in *26th IEEE International Conference on Computer Communications*, pp. 1973 –1981, May 2007.
- [27] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiawicz, “Tapestry: a resilient global-scale overlay for service deployment,” *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41 – 53, 2004.
- [28] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware ’01, (London, UK), pp. 329–350, Springer-Verlag, 2001.

-
- [29] M. Datar, “Butterflies and Peer-to-Peer Networks,” in *Proceedings of the 10th Annual European Symposium on Algorithms*, ESA '02, (London, UK), pp. 310–322, Springer-Verlag, 2002.
 - [30] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Looking up data in P2P systems,” *Communications of the ACM*, vol. 46, pp. 43–48, February 2003.
 - [31] National Institute of Standards and Technology, U.S. Department of Commerce, “Federal Information Processing Standard (FIPS) Publication 180-1, “Secure Hash Standard””, April 1995.
 - [32] M. Srivatsa and L. Liu, “Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis,” in *Proceedings of the 20th Annual Computer Security Applications Conference*, (Washington, DC, USA), pp. 252–261, IEEE Computer Society, 2004.
 - [33] K. R. B. Butler, S. Ryu, P. Traynor, and P. D. McDaniel, “Leveraging Identity-Based Cryptography for Node ID Assignment in Structured P2P Systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 12, pp. 1803–1815, 2009.
 - [34] T. Stading, “Secure Communication in a Distributed System Using Identity Based Encryption,” in *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 414–420, May 2003.
 - [35] K. V. Nguyen, “Simplifying Peer-to-Peer Device Authentication Using Identity-Based Cryptography,” in *Proceedings of the International conference on Networking and Services*, (Washington, DC, USA), p. 43, 2006.

-
- [36] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, “The Peersim simulator.” <http://peersim.sf.net>, Last visited on January 2011.
- [37] “Chord Implementation.” <http://peersim.sourceforge.net/>, Last visited on January 2011.
- [38] “MIRACL: Multiprecision Integer and Rational Arithmetic C/C++ Library.” <http://www.shamus.ie/>, Version 5.4.3, Last visited on January 2011.
- [39] “Skype.” <http://www.skype.com/>, Last visited on January 2011.

Glossary

AES Advanced Encryption Standard. 10, 42, 44, 48, 49, 51–53, 56

BDH Bilinear Diffie-Hellman. 16

CA Certificate Authority. 3, 4, 11, 13

DHT Distributed Hash Table. 25, 27

DLL Dynamic Link Library. 56

ECC Elliptic Curve Cryptography. 13

IBE Identity-based Encryption. iii, 4, 5, 9, 11–17, 19, 32–36, 38–44, 46, 48–54, 56, 58, 61, 63, 71, 73, 74

MIRACL Multi-precision Integer and rational Arithmetic Library. 55

MITM Man in the Middle. 31

P2P Peer-to-Peer. iii, 1–9, 21–25, 30–40, 42, 47, 55, 56, 72–76

PKC Public Key Cryptography. iii, 3, 4, 6, 10–13, 15, 33, 50, 51

PKG Private Key Generator. 4, 5, 12, 14, 15, 17, 20

RA Revocation Authority. 4

RSA Rivest, Shamir and Adleman. 11

SHA Secure Hash Algorithm. 25

TAG Trusted Authority Gateway. 37

TTL Time-To-Live. 24